

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

WEST Search History

[Hide Items](#)[Restore](#)[Clear](#)[Cancel](#)

DATE: Friday, October 15, 2004

Hide?	Set Name	Query	Hit Count
		<i>DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ</i>	
<input type="checkbox"/>	L9	L8 and I1	1
<input type="checkbox"/>	L8	L5 and ((maximum near number) same register)	85
<input type="checkbox"/>	L7	L6 and L1	3
<input type="checkbox"/>	L6	L5 and (maximum near number)	184
<input type="checkbox"/>	L5	L3	442
		<i>DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ</i>	
<input type="checkbox"/>	L4	L3	2
		<i>DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ</i>	
<input type="checkbox"/>	L3	I2 and allocat\$	444
<input type="checkbox"/>	L2	scratch near3 register	969
		<i>DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ</i>	
<input type="checkbox"/>	L1	717/124-135,145-147,151-161.ccls.	2887









END OF SEARCH HISTORY

[Web](#)
[Images](#)
[Directory](#)
[Local](#)
[NEW!](#)
[News](#)
[Products](#)

YAHOO! search

[Shortcuts](#)
[Advanced Search](#)
[Preferences](#)

Search Results Results **11 - 20** of about **8,330** for **"maximum number" scratch register allocation** - 0.17 sec. (Abo

11. <http://www.cs.umb.edu/~bill/java/tools/gnu/gcc-2.8.1/info/gcc.info-21> 
 This is Info file gcc.info, produced by Makeinfo version 1.67 from the input file gcc.texi. This file documents the use a
www.cs.umb.edu/~bill/java/tools/gnu/gcc-2.8.1/info/gcc.info-21 - 47k - [Cached](#) - [More from this site](#)
12. [Presented by Stephen Hines Presented by Stephen Hines COP 5622: Advanced Topics in Comp](#)
 ... necessary to store live **scratch** registers before calls and ... as number of registers needed for **register allocatio**
ww2.cs.fsu.edu/~hines/present/regalloc/regalloc.pdf - 359k - [View as html](#) - [More from this site](#)
13. <http://www.tochna.technion.ac.il/project/assert/gcc-2.7.1/flow.c> 
 /* Data flow analysis for GNU compiler.
www.tochna.technion.ac.il/project/assert/gcc-2.7.1/flow.c - 95k - [Cached](#) - [More from this site](#)
14. <http://www.cs.umb.edu/~bill/java/tools/gnu/gcc-3.0/info/gcc.info-28> 
 Published by the Free Software Foundation 59 Temple Place - Suite 330 Boston, MA 02111-1307 USA Copyright (C
 Inc.
www.cs.umb.edu/~bill/java/tools/gnu/gcc-3.0/info/gcc.info-28 - 48k - [Cached](#) - [More from this site](#)
15. <http://www.watson.org/~arr/sri-audit/src/contrib/gcc/reload1.c> 
 /* Reload pseudo regs into hard regs for insns that require hard regs.
www.watson.org/~arr/sri-audit/src/contrib/gcc/reload1.c - 244k - [Cached](#) - [More from this site](#)
16. <http://wwwcsif.cs.ucdavis.edu/~farrens/academic/250A/simulators/simplesim.dec/info/gcc.info-19>
 This is Info file gcc.info, produced by Makeinfo-1.63 from the input file gcc.texi. This file documents the use and the i
wwwcsif.cs.ucdavis.edu/~farrens/academic/250A/info/gcc.info-19 - 51k - [Cached](#) - [More from this site](#)
17. <http://docs.freebsd.org/doc/2.0.5-RELEASE/usr/share/info/gcc.info-19> 
 This is Info file gcc.info, produced by Makeinfo-1.55 from the input file /usr/src/gnu/usr.bin/cc/doc/gcc.texi. This file d
docs.freebsd.org/doc/2.0.5-RELEASE/usr/share/info/gcc.info-19 - 51k - [Cached](#) - [More from this site](#)
18. <http://www.eecs.harvard.edu/~nr/toolkit/working/derive/derive.orig/src/solve-reg.c> 
 else { nbytes = spec.nbytes = n/ninst; demand(nbytes <= sizeof (bigint), will exceed inst); } fixendian(binary, n, nbytes
[eecs.harvard.edu/~nr/toolkit/working/derive/derive.orig/src/solve-reg.c](http://www.eecs.harvard.edu/~nr/toolkit/working/derive/derive.orig/src/solve-reg.c) - 15k - [Cached](#) - [More from this site](#)
19. <http://www.uni-duesseldorf.de/ftp/share/egcs-1.0/info/gcc.info-21> 
 This is Info file gcc.info, produced by Makeinfo version 1.67 from the input file /gcc.texi. This file documents the use
www.uni-duesseldorf.de/ftp/share/egcs-1.0/info/gcc.info-21 - 50k - [Cached](#) - [More from this site](#)
20. <http://cs.nyu.edu/~yap/unsup/unsup/installers/exact/gcc/gcc-3.1/gcc/config/fr30/fr30.h> 
 ... define COMPILER_SCRATCH_REGISTER 0 /* The register that contains ... desirable to choose register alloca
cs.nyu.edu/~yap/unsup/unsup/installers/exact/gcc/gcc-3.1/gcc/config/fr30/fr30.h - 64k - [Cached](#) - [More from this site](#)

Results Page:

[Prev](#)
[◀](#)
[1](#)
[2](#)
[3](#)
[4](#)
[5](#)
[6](#)
[7](#)
[8](#)
[9](#)
[10](#)
[▶](#)
[Next](#)

[Web](#)
[Images](#)
[Directory](#)
[Local](#)
[NEW!](#)
[News](#)
[Products](#)

Your Search:

Help us improve your search experience. [Send us feedback.](#)

Create your own personal search experience with [My Yahoo! Search \[BETA\]](#)

Copyright © 2004 Yahoo! Inc. All rights reserved. [Privacy Policy](#) - [Terms of Service](#) - [Submit Your Site](#)



[Web](#) [Images](#) [Groups](#) [News](#) [Froogle](#) [more »](#)

"maximum number" scratch register allocation

[Search](#)

[Advanced Search](#)
[Preferences](#)

Web

Results 1 - 10 of about 3,490 for "**maximum number**" **scratch register allocation**. (0.33 seconds)

Free Pascal Programmers' manual

... MAXFPUREGISTERS : Maximum number of FPU ... float result register self register frame
pointer register stack pointer register scratch registers Processor ...
www.freepascal.org/docs-html/prog/prog.html - 49k - [Cached](#) - [Similar pages](#)

[PDF] Register Allocation by Priority-based Coloring

File Format: PDF/Adobe Acrobat - [View as HTML](#)

... be necessary to store live scratch registers before ... as number of registers needed
for register allocation approaches the maximum number of registers. ...
ww2.cs.fsu.edu/~hines/present/regalloc/regalloc.pdf - [Similar pages](#)

Register Classes

... is used for the class of the scratch register. ... such a definition would be to slow
down register allocation. ... AC expression for the maximum number of consecutive ...
www.emerson.emory.edu/services/gcc/html/Register_Classes.html - 21k - [Cached](#) - [Similar pages](#)

Small cleanup wrt. scratch allocation

... extern int max_regno; /* Maximum number of SCRATCH rx's ... Register information indexed
by register number */ typedef ... scratch allocation: From: Jeffrey A Law, ...
gcc.gnu.org/ml/gcc-patches/1998-10/msg00656.html - 14k - [Cached](#) - [Similar pages](#)

Patch to delete scratch allocation in local-alloc

... which may lead to suboptimal register allocation. ... This sets the maximum number of
quantities we ... mark_scratch_live (scratch) - rx scratch; {- register int i ...
gcc.gnu.org/ml/gcc-patches/1998-10/msg00059.html - 18k - [Cached](#) - [Similar pages](#)
[\[More results from gcc.gnu.org \]](#)

Using and Porting GNU CC - Target Description Macros

... MAX_WCHAR_TYPE_SIZE Maximum number for the size in bits ... that take place after register
allocation and could ... memory, but require a scratch register for stores ...
www.math.utah.edu/docs/info/gcc_19.html - 101k - [Cached](#) - [Similar pages](#)

[PS] Allocation of Global Data Objects in On-Chip RAM

File Format: Adobe PostScript - [View as Text](#)

... Memory allocation in scratch-pad memory has been ... be generated from spilling after
register allocation are not ... last column shows the maximum number of onchip ...
user.it.uu.se/~jans/papers/global.ps - [Similar pages](#)

Using and Porting GNU CC - Target Description Macros

... MAX_WCHAR_TYPE_SIZE Maximum number for the size in bits of the ... It can be used for
allocation within a ... from memory, but require a scratch register for stores to ...
www.ia.pw.edu.pl/~wujek/dokumentacja/gnu/gcc/gcc_17.html - 101k - [Cached](#) - [Similar pages](#)

[PDF] Methods for Saving and Restoring Register Values across Function ...

File Format: PDF/Adobe Acrobat - [View as HTML](#)

... Six existing registers, the maximum number of registers that could be ... that was associated
with a non-existent register. ... and the number of scratch registers for ...
www.cs.ubc.ca/local/reading/proceedings/spe91-95/spe/vol21/issue2/spe009jd.pdf - [Similar pages](#)

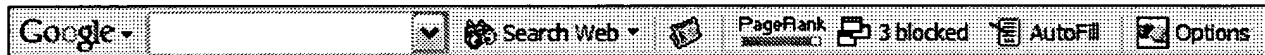
Sc: Simd within a register C Compiler

... edge." It was created from scratch using PCCTS... the pipelined code-schedule and register allocation for each ... allows you to specify the maximum number of seconds ...
dynamo.ecn.purdue.edu/~hankd/SWAR/Sc.html - 34k - [Cached](#) - [Similar pages](#)

Google

Result Page: 1 2 3 4 5 6 7 8 9 10 [Next](#)

Free! Get the Google Toolbar. [Download Now](#) - [About Toolbar](#)



"maximum number" scratch register [Search](#)

[Search within results](#) | [Language Tools](#) | [Search Tips](#) | [Dissatisfied? Help us improve](#)

[Google Home](#) - [Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2004 Google



Terms used **scratch register** AND **allocation**

Found **14,838** of **143,484**

Sort results
by



[Save results to a Binder](#)

Try an [Advanced Search](#)

Display
results



[Search Tips](#)

Try this search in [The ACM Guide](#)

☐ Open results in a new
window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

1 [Fast, effective code generation in a just-in-time Java compiler](#)

Ali-Reza Adl-Tabatabai, Michał Cierniak, Guei-Yuan Lueh, Vishesh M. Parikh, James M. Stichnoth

May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation**, Volume 33 Issue 5

Full text available: pdf(1.44 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A "Just-In-Time" (JIT) Java compiler produces native code from Java byte code instructions during program execution. As such, compilation speed is more important in a Java JIT compiler than in a traditional compiler, requiring optimization algorithms to be lightweight and effective. We present the structure of a Java JIT compiler for the Intel Architecture, describe the lightweight implementation of JIT compiler optimizations (e.g., common subexpression elimination, register allocation, and elim ...

2 [The priority-based coloring approach to register allocation](#)

Fred C. Chow, John L. Hennessy

October 1990 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 12 Issue 4

Full text available: pdf(2.97 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Global register allocation plays a major role in determining the efficacy of an optimizing compiler. Graph coloring has been used as the central paradigm for register allocation in modern compilers. A straightforward coloring approach can suffer from several shortcomings. These shortcomings are addressed in this paper by coloring the graph using a priority ordering. A natural method for dealing with the spilling emerges from this approach. The detailed algorithms for a priority-based colori ...

3 [Compilers II: Inter-procedural stacked register allocation for itanium® like architecture](#)

Liu Yang, Sun Chan, G. R. Gao, Roy Ju, Guei-Yuan Lueh, Zhaoqing Zhang

June 2003 **Proceedings of the 17th annual international conference on Supercomputing**

Full text available: pdf(478.20 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

A hardware managed register stack, Register Stack Engine (RSE), is implemented in Itanium® architecture to provide a unified and flexible register structure to software. The compiler allocates each procedure a register stack frame with its size explicitly specified using an *alloc* instruction. When the total number of registers used by the procedures on the

call stack exceeds the number of physical registers, RSE performs automatic register overflows and fills to ensure that the c ...

Keywords: hot region, hotspot, inter-procedural stacked register allocation, quota assignment, register allocation

4 Towards a family of languages for the design and implementation of machine architectures

Subrata Dasgupta, Marius Olafsson

April 1982 **Proceedings of the 9th annual symposium on Computer Architecture**

Full text available:  pdf(759.02 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In recent years, increases in complexity of hardware/firmware systems, and the concern for systems reliability have resulted in growing interest in methodologies and tools for the design, description and verification of computer systems. A vital component of any such design methodology is the language used for representing the design. In the case of particularly complex systems the design process may involve a succession of stages each of which represents the system at a particular level of ...

5 Hot cold optimization of large Windows/NT applications

Robert Cohn, P. Geoffrey Lowney

December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:  pdf(1.14 MB)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)
[Publisher Site](#)

A dynamic instruction trace often contains many unnecessary instructions that are required only by the unexecuted portion of the program. Hot-cold optimization (HCO) is a technique that realizes this performance opportunity. HCO uses profile information to partition each routine into frequently executed (hot) and infrequently executed (cold) parts. Unnecessary operations in the hot portion are removed, and compensation code is added on transitions from hot to cold as needed. We evaluate HCO on a ...

Keywords: optimization, profile, NT, register allocation

6 A practical method for code generation based on exhaustive search

David W. Krumme, David H. Ackley

June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6

Full text available:  pdf(1.10 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

An original method for code generation has been developed in conjunction with the construction of a compiler for the C programming language on the DEC-10 computer. The method is comprehensive, determining evaluation order and doing register allocation and instruction selection simultaneously. It uses exhaustive search rather than heuristics, and is table-driven, with most machine-specific information isolated in the tables. Testing and evaluation have shown that the method is effective, the ...

7 Cray Pascal

N. H. Madhavji, I. R. Wilson

June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6

Full text available:  [pdf\(731.22 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper presents an investigation of the design decisions taken in the implementation of a compiler for Pascal on the CRAY-1 computer. The structured nature of Pascal statements and data structures is contrasted with the 'powerful computing engine' nature of the CRAY-1 hardware. The accepted views of Pascal as a simple one-pass language and the CRAY-1 as a vector processor are laid aside in favour of a multi-pass approach, taking account of the machine's scalar capabilities. The project ...

Keywords: CRAY-1, Code optimisation, Compilation, Pascal, Vector processors

8 Porting the Zed compiler

G. B. Bonkowski, W. M. Gentleman, M. A. Malcolm

August 1979 **ACM SIGPLAN Notices , Proceedings of the 1979 SIGPLAN symposium on Compiler construction**, Volume 14 Issue 8

Full text available:  [pdf\(604.77 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Zed is the base language used to implement the portable realtime operating system Thoth [7], and to write commands, utilities, application programs, and other software which run under Thoth. (Zed is similar to C, although language details are not important in this paper.) One of the founding principles of Thoth is our experience that the hardest problems in porting programs usually arise when interfacing to different operating systems. By porting the whole operating system first, we ensure t ...

9 16-bit vs. 32-bit instructions for pipelined microprocessors

John Bunda, Don Fussell, W. C. Athas, Roy Jenevein

May 1993 **ACM SIGARCH Computer Architecture News , Proceedings of the 20th annual international symposium on Computer architecture**, Volume 21 Issue 2


Full text available:  [pdf\(883.21 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In any stored-program computer system, information is constantly transferred between the memory and the instruction processor. Machine instructions are a major portion of this traffic. Since transfer bandwidth is a limited resource, inefficiency in the encoding of instruction information (low code density) can have definite hardware and performance costs. Starting with a parameterized baseline RISC design, we compare performance for two instruction encodings for the same instruct ...

10 A retargetable register allocation framework for embedded processors

Jean-Marc Daveau, Thomas Thery, Thierry Lepley, Miguel Santana

June 2004 **ACM SIGPLAN Notices , Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools**, Volume 39 Issue 7

Full text available:  [pdf\(1.03 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper describes the FlexCC2 register allocation framework. FlexCC2 is an optimizing retargetable C compiler for embedded processors, and in particular for DSP processors. Embedded processors often contain features such as irregular and constrained register sets that complicate register allocation, making traditional methods inefficient. In this paper, we present a register allocation framework specifically tailored for embedded processor specificities. This framework has been integrated in ...

Keywords: embedded processors, register allocation

Engineering a production code generator

John Crawford

June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction**, Volume 17 Issue 6


Full text available:  [pdf\(875.84 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes the structure of a code generator formed by merging the best aspects of three code generation techniques: Graham-Glanville parser-driven code generation [G] [GG] [GR] [HG], the register allocation/spill mechanism from the Portable C Compiler [J], and a code template expander [W]. The Graham-Glanville method was modified to use a standard LALR parser and table builder, and the register allocation method was extended in several significant ways in order to make optimal us ...

12 ASHs: Application-specific handlers for high-performance messaging

Deborah A. Wallach, Dawson R. Engler, M. Frans Kaashoek

August 1996 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Applications, technologies, architectures, and protocols for computer communications**, Volume 26 Issue 4

Full text available:  [pdf\(174.50 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Application-specific safe message handlers (ASHs) are designed to provide applications with hardware-level network performance. ASHs are user-written code fragments that safely and efficiently execute in the kernel in response to message arrival. ASHs can direct message transfers (thereby eliminating copies) and send messages (thereby reducing send-response latency). In addition, the ASH system provides support for dynamic integrated layer processing (thereby eliminating duplicate message ...

13 Application specific processors: A low power architecture for embedded perception

Binu Mathew, Al Davis, Mike Parker

September 2004 **Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems**

Full text available:  [pdf\(310.49 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Recognizing speech, gestures, and visual features are important interface capabilities for future embedded mobile systems. Unfortunately, the real-time performance requirements of complex perception applications cannot be met by current embedded processors and often even exceed the performance of high performance microprocessors whose energy consumption far exceeds embedded energy budgets. Though custom ASICs provide a solution to this problem, they incur expensive and lengthy design cycles and ...

Keywords: VLIW, computer vision, embedded systems, low power design, perception, speech recognition, stream processor

14 A Fortran compiler for the FPS-164 scientific computer

Roy F. Touzeau

June 1984 **ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium on Compiler construction**, Volume 19 Issue 6

Full text available:  [pdf\(863.50 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#)

15 Dynamic analysis: The design and implementation of FIT: a flexible instrumentation toolkit

Bruno De Bus, Dominique Chagnet, Bjorn De Sutter, Ludo Van Put, Koen De Bosschere

Full text available:  [pdf\(252.10 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper presents FIT, a Flexible open-source binary code Instrumentation Toolkit. Unlike existing tools, FIT is truly portable, with existing backends for the Alpha, x86 and ARM architectures and the Tru64Unix, Linux and ARM Firmware execution environments. This paper focuses on some of the problems that needed to be addressed for providing this degree of portability. It also discusses the trade-off between instrumentation precision and low overhead.

Keywords: code compaction, performance code abstraction

16 Compiling Prolog into microcode: a case study using the NCR/32-000

B. Fagin, Y. N. Patt, V. Srin, A. Despain

December 1985 **ACM SIGMICRO Newsletter , Proceedings of the 18th annual workshop on Microprogramming**, Volume 16 Issue 4

Full text available:  [pdf\(1.01 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A proven method of obtaining high performance for Prolog programs is to first translate them into the instruction set of Warren's Abstract Machine, or W-code [1]. From that point, there are several models of execution available. This paper describes one of them:- the compilation of W-code directly into the vertical microcode of a general purpose host processor, the NCR/32-000. The result is the fastest functioning Prolog system known to the authors. We describe the implementation, provide b ...

17 ASHs: application-specific handlers for high-performance messaging

Deborah A. Wallach, Dawson R. Engler, M. Frans Kaashoek

August 1997 **IEEE/ACM Transactions on Networking (TON)**, Volume 5 Issue 4

Full text available:  [pdf\(174.62 KB\)](#) Additional Information: [full citation](#), [references](#), [index terms](#)

Keywords: computer networks, dynamic code generation, modular computer systems, operating systems, protocols, software protection, user-level networking

18 Packet types: abstract specification of network protocol messages

Peter J. McCann, Satish Chandra

August 2000 **ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication**, Volume 30 Issue 4



Full text available:  [pdf\(435.48 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

In writing networking code, one is often faced with the task of interpreting a raw buffer according to a standardized packet format. This is needed, for example, when monitoring network traffic for specific kinds of packets, or when unmarshaling an incoming packet for protocol processing. In such cases, a programmer typically writes C code that understands the grammar of a packet and that also performs any necessary byte-order and alignment adjustments. Because of the complexity of certain ...

19 Evaluation of scheduling techniques on a SPARC-based VLIW testbed

Seongbae Park, SangMin Shim, Soo-Mook Moon

December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**

Full text available:  pdf(1.40 MB)  Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)
[Publisher Site](#)

The performance of Very Long Instruction Word (VLIW) microprocessors depends on the close cooperation between the compiler and the architecture. This paper evaluates a set of important compilation techniques and related architectural features for VLIW machines. The evaluation is performed on a SPARC-based VLIW testbed where gcc-generated optimized SPARC code is scheduled into high-performance VLIW code. As a base scheduling compiler, we experiment with three core scheduling techniques including ...

Keywords: SPARC-based VLIW testbed, VLIW microprocessors, Very Long Instruction Word microprocessors, all-path speculation, compiler, computer architecture, copies, gcc-generated optimized SPARC code, high-performance VLIW code, loop unrolling, memory disambiguation, nongreedy enhanced pipeline scheduling, nonspeculative operations, parallel machines, performance, profile-based all-path speculation, renaming, restricted speculative loads, scheduling compiler, scheduling techniques, software pipelining, speculative operations, trace-based speculation

20 [Direct execution models of processor behavior and performance](#)

Richard M. Fujimoto, William B. Campbell

December 1987 **Proceedings of the 19th conference on Winter simulation**

Full text available:  pdf(952.49 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper discusses a modeling technique for creating efficient instruction level simulation models of von Neumann processors. In contrast to traditional approaches which use a software interpreter, this technique employs direct execution of application programs on the host computer. An assembly language program for the target machine is decompiled to a high level language, instrumented, and then recompiled and executed on the host computer. A prototype im ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)

Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced
- ☐ CrossRef

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

IEEE Enterprise

- ☐ Access the IEEE Enterprise File Cabinet

 Print Format
Your search matched **5** of **1079782** documents.A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:

You may refine your search by editing the current search expression or entering a new one in the text box.

☐ Check to search within this result set

Results Key:

JNL = Journal or Magazine **CNF** = Conference **STD** = Standard**1 On the problem of in-place calculating LPC parameters***Xixian Chen; Weixin Li;*

Circuits and Systems, 1991., IEEE International Symposium on , 11-14 June 1991

Pages:316 - 319 vol.1

[\[Abstract\]](#) [\[PDF Full-Text \(244 KB\)\]](#) IEEE CNF**2 Automatic categorization algorithm for evolvable software archive***Kawaguchi, S.; Garg, P.K.; Matsushita, M.; Inoue, K.;*

Software Evolution, 2003. Proceedings. Sixth International Workshop on Principles of , 1-2 Sept. 2003

Pages:195 - 200

[\[Abstract\]](#) [\[PDF Full-Text \(282 KB\)\]](#) IEEE CNF**3 Conflict-free access to multiple single-ported register files***Mueller, S.M.; Vishkin, U.;*

Parallel Processing Symposium, 1997. Proceedings., 11th International , 1-5 April 1997

Pages:672 - 678

[\[Abstract\]](#) [\[PDF Full-Text \(560 KB\)\]](#) IEEE CNF**4 Assessment design for mathematics web project-based learning***Yuen Chen; I-hua Lin; Yann-jiun Tzeng; Pi-hsia Hung; Weichung Wang;*

Computers in Education, 2002. Proceedings. International Conference on , 3-6 Dec. 2002

Pages:1329 - 1330 vol.2

[\[Abstract\]](#) [\[PDF Full-Text \(327 KB\)\]](#) IEEE CNF

5 Reverse compilation of digital signal processor assembler source to ANSI-C

Johnstone, A.; Scott, E.; Womack, T.;

Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on , 30 Aug.-3 Sept. 1999

Pages:316 - 325

[\[Abstract\]](#) [\[PDF Full-Text \(184 KB\)\]](#) **IEEE CNF**

[Home](#) | [Log-out](#) | [Journals](#) | [Conference Proceedings](#) | [Standards](#) | [Search by Author](#) | [Basic Search](#) | [Advanced Search](#) | [Join IEEE](#) | [Web Account](#) | [New this week](#) | [OPAC Linking Information](#) | [Your Feedback](#) | [Technical Support](#) | [Email Alerting](#) | [No Robots Please](#) | [Release Notes](#) | [IEEE Online Publications](#) | [Help](#) | [FAQ](#) | [Terms](#) | [Back to Top](#)

Copyright © 2004 IEEE — All rights reserved